

# Fast multivariate log-concave density estimation

Fabian Rathke<sup>a,\*</sup>, Christoph Schnörr<sup>a</sup>

<sup>a2</sup>*Image & Pattern Analysis Group (IPA), Heidelberg University, Im Neuenheimer Feld 205, 69120 Heidelberg, Germany.*

---

## Abstract

A novel computational approach to log-concave density estimation is proposed. Previous approaches utilize the piecewise-affine parametrization of the density induced by the given sample set. The number of parameters as well as non-smooth subgradient-based convex optimization for determining the maximum likelihood density estimate cause long runtimes for dimensions  $d \geq 2$  and large sample sets. The presented approach is based on mildly non-convex smooth approximations of the objective function and *sparse*, adaptive piecewise-affine density parametrization. Established memory-efficient numerical optimization techniques enable to process larger data sets for dimensions  $d \geq 2$ . While there is no guarantee that the algorithm returns the maximum likelihood estimate for every problem instance, we provide comprehensive numerical evidence that it does yield near-optimal results after significantly shorter runtimes. For example, 10000 samples in  $\mathbb{R}^2$  are processed in two seconds, rather than in  $\approx 14$  hours required by the previous approach to terminate. For higher dimensions, density estimation becomes tractable as well: Processing 10000 samples in  $\mathbb{R}^6$  requires 35 minutes. The software is publicly available as CRAN R package `fmlogcondens`.

Keywords: log-concavity, maximum likelihood estimation, nonparametric density estimation, adaptive piecewise-affine parametrization

---

## 1. Introduction

### 1.1. Motivation, Related Work

Log-concave density estimation has been an active area of research. Quoting [Chen and Samworth \(2013\)](#), the “allure is the prospect of obtaining fully automatic nonparametric estimators, with no tuning parameters to choose”, as a flexible alternative to parametric models, like the Gaussian, that are often adopted by practitioners in an ad-hoc way. The mathematical analysis as well as the design of algorithms benefit from the convexity properties of the class of log-concave densities. We refer to [Samworth \(2017\)](#) for a recent survey.

The general form of a log-concave density reads

$$f(x) = \exp(-\varphi(x)), \quad \varphi \in \mathcal{F}_0(\mathbb{R}^d), \quad (1.1)$$

where  $\mathcal{F}_0(\mathbb{R}^d)$  denotes the class of convex lower-semicontinuous proper functions  $\varphi: \mathbb{R}^d \rightarrow (-\infty, \infty]$  such that  $\int_{\mathbb{R}^d} f = 1$ . Given i.i.d. samples

$$\mathcal{X}_n = \{x_1, \dots, x_n\} \subset \mathbb{R}^d \quad (1.2)$$

of a random vector  $X \sim f$ , with  $n \geq d + 1$ , the task is to determine an estimate

$$\hat{f}_n = \exp(-\hat{\varphi}_n(x)) \quad (1.3)$$

of  $f$ . This estimate, determined as maximizer of the log-likelihood, exists and is unique with probability 1 ([Cule et al., 2010](#), Thm. 1). Moreover, the corresponding convex function  $\hat{\varphi}_n \in \mathcal{F}_0$  is supported on the

---

\*Corresponding author (Mail: [fabian.rathke@iwr.uni-heidelberg.de](mailto:fabian.rathke@iwr.uni-heidelberg.de)/frathke@gmail.com, Tel.: +49 6221 5414858)

convex hull  $C_n = \text{conv } \mathcal{X}_n$  of the given data and is *piecewise linear*, in the sense of [Rockafellar and Wets \(2009, Def. 2.47\)](#):  $C_n$  can be represented as union of finitely many polyhedral sets

$$C_n = \bigcup_{i=1}^{N_{n,d}} C_{n,i}, \quad (1.4)$$

relative to each of which  $\hat{\varphi}_n$  admits the *affine* representation

$$\hat{\varphi}_n(x)|_{C_{n,i}} =: \hat{\varphi}_{i,n}(x) = \langle a_i, x \rangle + b_i, \quad a_i \in \mathbb{R}^d, b_i \in \mathbb{R}, \quad i = 1, \dots, N_{n,d}. \quad (1.5)$$

This is equivalent to the fact that the epigraph of  $\hat{\varphi}_n$ , denoted by

$$\text{epi } \hat{\varphi}_n = \{(x, \alpha) \in \mathbb{R}^d \times \mathbb{R} : \alpha \geq \hat{\varphi}_n(x)\} \quad (1.6)$$

is polyhedral and  $\hat{\varphi}_n$  admits the representation ([Rockafellar and Wets, 2009, Thm. 2.49](#))

$$\hat{\varphi}_n = \begin{cases} \max \{ \hat{\varphi}_{1,n}(x), \dots, \hat{\varphi}_{N_{n,d},n}(x) \}, & x \in C_n, \\ \infty, & x \notin C_n. \end{cases} \quad (1.7)$$

We denote the class of piecewise linear proper convex functions over  $C_n$  by

$$\Phi_n := \{ \varphi_n \in \mathcal{F}_0(\mathbb{R}^d) : \varphi_n \text{ has the form (1.5) and (1.7)} \}. \quad (1.8)$$

Figure 1.1 displays a function  $\varphi_n$  in the planar case  $d = 2$ . Given the function values

$$y_\varphi = (y_{\varphi,1}, \dots, y_{\varphi,n}) := (\varphi_n(x_1), \dots, \varphi_n(x_n)), \quad (1.9)$$

$\varphi_n$  is uniquely determined as *lower convex envelope*, that is the largest convex function majorized at the given sample points  $x_i$  by  $y_{\varphi,i}$ ,

$$\varphi_n(x_i) \leq y_{\varphi,i}, \quad i = 1, \dots, n. \quad (1.10)$$

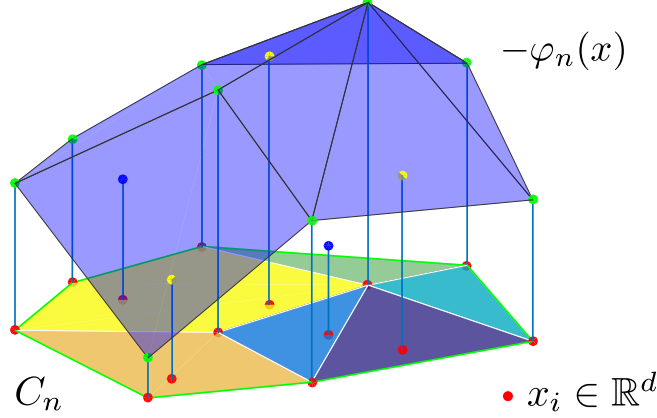
Due to [Cule et al. \(2010, Thm. 2\)](#), a natural and admissible variational approach for determining the maximum likelihood estimate  $\hat{f}_n$  in terms of  $\hat{\varphi}_n$  and  $\hat{y}_\varphi$ , respectively, is given by

$$\hat{y}_\varphi = \arg \min_{y_\varphi} J(y_\varphi), \quad J(y_\varphi) = \frac{1}{n} \sum_{i=1}^n y_{\varphi,i} + \int_{C_n} \exp(-\varphi_n(x)) \, dx \quad (1.11)$$

where the latter integral acts like a Lagrangian multiplier term enforcing the constraint  $\int_{C_n} f_n = 1$  ([Silverman, 1982, Thm. 3.1](#)). In fact, it was shown that solving problem (1.11) amounts to effectively minimizing over  $\Phi_n$  (1.8) to obtain  $\hat{\varphi}_n$  and in turn the ML-estimate (1.3).

An algorithm for computing  $\hat{y}_\varphi$  was worked out by [Cule et al. \(2010\)](#) based on the convexity of  $J$ . While this algorithm is guaranteed to return the global optimum, its runtime complexity suffers from two facts:

- (i) The objective function  $J$  is convex but *non-smooth* due to the polyhedral class of functions (1.8) in which the algorithm searches for  $\hat{\varphi}_n$ . As consequence, the iterative scheme is based on subgradients which are known to converge rather slowly.
- (ii) The integral of (1.11) has to be evaluated in every iterative step for each subset  $C_{n,i}$  of the polyhedral decomposition (1.4), where the subsets  $C_{n,i}$  are required to be *simplices*. While this can be conveniently done in closed form ([Cule et al., 2010, App. B](#)), it is the increasing number of these subsets for larger dimension  $d > 2$  that slows down the algorithm.

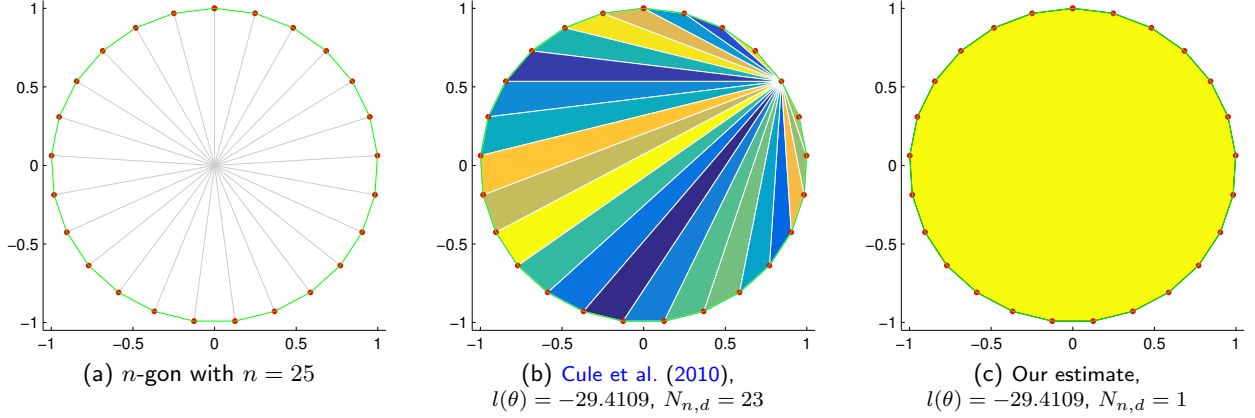


**Figure 1.1:** A piecewise affine concave function  $-\varphi_n(x)$  (1.5) whose parameters determine the density estimate (1.3). The function values  $-\varphi(x_1), \dots, -\varphi(x_n)$  at given data points  $\mathcal{X}_n = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$  induce a polyhedral decomposition  $C_n$  of the convex hull  $\text{conv}(X)$ , here shown for the case of bivariate data. Determining the parameters through iterative numerical optimization may change this decomposition depending on which data point defines a vertex of the hypograph (green point) or not (yellow and blue points). This increases the complexity considerably, in particular for larger  $n$  and dimension  $d$ . In this paper, we work with a smooth approximation that enables more efficient log-concave density estimation.

The number  $N_{n,d}$  of components of the decomposition (1.4) is known to depend linearly on  $n$ ,  $N_{n,d} = \mathcal{O}(n)$ , for  $n$  points *uniformly* distributed in  $\mathbb{R}^d$  (Dwyer, 1991), whereas the worst case bound for ‘pathologically’ distributed  $n$  points is  $N_{n,d} = \mathcal{O}(n^{\lceil \frac{d}{2} \rceil})$ , i.e. grows exponentially with the dimension  $d$  (McMullen, 1970). For  $n$  points sampled from log-concave distributions that are unimodal and in this sense simply shaped, it is plausible to assume that the lower complexity bound holds approximately, i.e. a *linear dependency*  $N_{n,d} = \mathcal{O}(n)$ . This means, in particular, that the number of parameters of the affine functions forming  $\hat{\varphi}_n$  due to (1.7) linearly depends on  $n$  as well. While these bounds take into account the entire data set  $\mathcal{X}_n$ , it was shown for  $d = 1$  that under sufficient smoothness and other conditions, not all  $x_i$  need to participate in the decomposition  $C_n$  (Dümbgen and Rufibach, 2009). No proofs exist for  $d > 1$ , but results presented in this paper indicate that this property of the ML estimator carries over to the multivariate case. Therefore the actual dependency of  $N_{n,d}$  on  $n$  may be lower than  $\mathcal{O}(n)$ .

On the other hand, concerning the ultimate objective of accurately estimating a multivariate log-concave density  $f$ , it was recently shown by Diakonikolas et al. (2017) that in order to achieve an estimation error  $\epsilon$  in total variation distance with high probability, a function  $\hat{f}_n$  suffices that is defined by  $\mathcal{O}((1/\epsilon)^{(d+1)/2})$  hyperplanes. In the univariate case  $d = 1$ , an algorithm that matches this complexity bound was published recently (Acharya et al., 2017). In the multivariate case  $d > 1$ , on the other hand, the design of a computationally efficient algorithm was considered as a “challenging and important open question” by Diakonikolas et al. (2017).

Quite recently, two approaches were published (Axelrod and Valiant, 2018; Diakonikolas et al., 2018) which solve the log-concave MLE (1.11) with high probability with an estimation error  $\epsilon < 1$  in terms of the total log-likelihood in  $\text{poly}(n, d, 1/\epsilon)$  time. Both approaches are stochastic and rely on the work of Lovász and Vempala (2007) to sample from  $\varphi_n$  over the convex body  $C_n$ . Regarding the computational efficiency of the latter approach, Lovász and Deák (2012) noted that they “could not experiment with other convex bodies than cubes, because the oracle describing the convex bodies took too long to run”. Since neither Axelrod and Valiant (2018) nor Diakonikolas et al. (2018) provide an implementation of their novel approaches, a fair and competitive evaluation has to be left for future work.



**Figure 1.2:** Example due to D. Schuhmacher from the discussion in the appendix of Cule et al. (2010) regarding computational efficiency: (a)  $n=25$  data points  $\mathcal{X}_n$  form a  $n$ -gon. Due to the symmetry of  $\mathcal{X}_n$ , the density estimate  $\hat{f}_n$  is the uniform density (not explicitly shown here; both the approach Cule et al. (2010) and our approach return this estimate, as the equal log-likelihood values  $l(\theta)$  (3.1) demonstrate). It is clear that this uniform density can be represented by a *single* hyperplane,  $N_{n,d} = 1$ , that our approach correctly finds (panel (c)). In contrast, the approach of Cule et al. (b) relies on a *triangulation* of  $C_n$ , which leads to a more involved density parameter estimation problem:  $N_{n,d} = 23$  affine function parameters. This gap of complexity increases considerably with larger numbers  $n$  of data points and dimension  $d$  of the data space.

### 1.2. Contribution, Organization

This preceding discussion motivated us to address the two shortcomings (i), (ii) raised above as follows.

- (1) We consider the representation (1.8) of  $\hat{\varphi}_n$  and adopt a *smooth* approximation of the non-smooth max-operation. While the resulting modification of (1.11) no longer is convex, numerical methods can be applied that are orders of magnitude more efficient than the subgradient based iterative schemes of Cule et al. (2010). Furthermore, we exploit the fact that the smoothness of the approximation can be controlled by a single parameter  $\gamma$ : While we utilize strong smoothing to obtain an initial parameter vector, the subsequent optimization is carried out with minimal smoothing.
- (2) Rather than optimizing *all* parameters of (1.7), we apply a threshold criterion in order to drop ‘inactive’ hyperplanes, since the optimal estimate  $\hat{\varphi}_n$  can be expected to be defined by a small subset of them, as discussed above. This measure speeds up the computations too without essentially compromising the accuracy of the resulting density estimator  $\hat{f}_n$ . Moreover, unlike the approach of Cule et al. (2010), we do not restrict polyhedral subsets  $C_{n,i}$  to simplices. Figure 1.2 shows a somewhat extreme academical example in order to illustrate these points.

Due to the non-convexity of our objective function, we cannot guarantee that our approach determines the maximum-likelihood density estimate for *every* problem instance, as does the approach of Cule et al. (2010). This was the case, however, in a comprehensive series of numerical experiments indicate, that we report below. In particular, log-concave density estimation for large sample sets and for higher dimensions becomes computationally tractable.

Our paper is organized as follows. We present our approach in Section 2 and discuss details of the algorithm and its implementation. In Section 3, we report extensive numerical results up to dimension  $d = 6$  using sample sizes in the range  $n \in [10^2, 10^5]$ . In the univariate case  $d = 1$ , our method is on par with the active set approach of Dümbgen et al. (2007) regarding both runtime and accuracy. This method is not applicable to higher dimensions, however. In such cases,  $d \in \{2, \dots, 6\}$ , our method is as accurate as the algorithm of Cule et al. (2010) but orders of magnitude more efficient. For example, for  $d = 2$  and  $n = 10.000$  samples, the algorithm of Cule et al. takes 4.6 hours whereas our algorithm terminates after 0.5

seconds. For  $d = 6$  and  $n = 1.000$  samples, the algorithm of Cule et al. takes about 10 hours, whereas our algorithm terminates after 5 minutes.

An implementation of our approach is publicly available as software R package `fmlogcondens` (Rathke and Schnörr, 2018) on CRAN.

## 2. Approach

We define in Section 2.1 the objective function as smooth approximation of the negative log-likelihood. The subsequent sections discuss how the parameter values of the log-concave density estimate are determined by numerical optimization. The overall structure of the approach is summarized as Algorithm 1 on page 11.

### 2.1. Objective Function

We rewrite the negative log-likelihood functional (1.11) in the form

$$L(\theta) := \frac{1}{n} \sum_{i=1}^n \varphi_n(x_i) + \int_{C_n} \exp(-\varphi_n(x)) \, dx, \quad (2.1a)$$

$$\theta := \{(a_1, b_1), \dots, (a_{N_{n,d}}, b_{N_{n,d}})\}, \quad (2.1b)$$

where  $\varphi_n$  and all  $\varphi_{i,n}$  have the form (1.5) and (1.7), respectively, and  $\theta$  collects all parameters that determine  $\varphi_n$ . We define the log-concave density estimate (1.3) in terms of the function

$$\hat{\varphi}_n = \varphi_n|_{\theta=\hat{\theta}}: \quad \hat{\theta} \text{ locally minimizes } L(\theta). \quad (2.2)$$

Our next step is to smoothly approximate the representation (1.7) of  $\varphi_n$ . Using the convex log-exponential function

$$\text{logexp}: \mathbb{R}^d \rightarrow \mathbb{R}, \quad x \mapsto \text{logexp}(x) := \log \left( \sum_{i=1}^d e^{x_i} \right) \quad (2.3)$$

we introduce a *smoothing parameter*  $\gamma > 0$  and define the rescaled smooth convex function

$$\text{logexp}_\gamma: \mathbb{R}^d \rightarrow \mathbb{R}, \quad x \mapsto \text{logexp}_\gamma(x) := \gamma \log \exp \left( \frac{x}{\gamma} \right) = \gamma \log \left( \sum_{i=1}^d \exp \left( \frac{x_i}{\gamma} \right) \right), \quad (2.4)$$

that *uniformly* approximates the non-smooth max-operation (Rockafellar and Wets, 2009, Example 1.30) in the following sense:

$$\text{logexp}_\gamma(x) - \gamma \log d \leq \max_{i=1, \dots, d} \{x_1, \dots, x_d\} \leq \text{logexp}_\gamma(x), \quad \forall x \in \mathbb{R}^d. \quad (2.5)$$

Utilizing this function, we define in view of (1.7) the smooth approximation

$$\varphi_{n,\gamma}(x) := \begin{cases} \text{logexp}_\gamma(\varphi_{1,n}(x), \dots, \varphi_{N_{n,d},n}(x)), & x \in C_n, \\ \infty, & x \notin C_n, \end{cases} \quad (2.6)$$

and in turn the smooth approximation of the objective function (2.1)

$$L_\gamma(\theta) := \frac{1}{n} \sum_{i=1}^n \varphi_{n,\gamma}(x_i) + \int_{C_n} \exp(-\varphi_{n,\gamma}(x)) \, dx. \quad (2.7)$$

We point out that by virtue of (2.5), we have

$$\forall x \in C_n, \quad 0 \leq \varphi_{n,\gamma}(x) - \varphi_n(x) \leq \gamma \log d \rightarrow 0 \quad \text{for } \gamma \rightarrow 0 \quad (2.8)$$

and consequently, by continuity,

$$L_\gamma(\theta) \rightarrow L(\theta) \quad \text{for } \gamma \rightarrow 0. \quad (2.9)$$

## 2.2. Numerical Optimization

We apply an established, memory-efficient quasi-Newton method known as L-BFGS in the literature (Nocedal and Wright, 2006), to compute a sequence

$$(\theta^{(k)})_{k \geq 1} \quad (2.10)$$

of parameter values that converges to a local minimum  $\hat{\theta}$  of the objective function (2.7). A key aspect of this iterative procedure is to maintain at each step  $k$  an approximation  $H^{(k)}$  of the inverse Hessian  $(\nabla^2 L_\gamma(\theta^{(k)}))^{-1}$  of the objective function (2.7), in terms of a few gradients  $\nabla L_\gamma(\theta^{(k')})$  evaluated and stored at preceding iterative steps  $k' < k$ . This avoids to handle directly the Hessian of size  $(\dim \theta)^2 = ((d+1)N_{n,d})^2$  and hence enables to cope with much larger problem sizes.

The basic update steps with search direction  $p^{(k)}$  and step size  $\lambda_k$  read

$$\theta^{(k+1)} = \theta^{(k)} + \lambda_k p^{(k)}, \quad p^{(k)} = -H^{(k)} \nabla L_\gamma(\theta^{(k)}). \quad (2.11)$$

The stepsize  $\lambda_k$  is determined by backtracking line search. More specifically, we select the largest  $\lambda_k$  in the set  $\{1, p, p^2, \dots\}$ ,  $p = 0.1$ , such that the condition

$$L_\gamma(\theta^{(k)} + \lambda_k p^{(k)}) - L_\gamma(\theta^{(k)}) \leq \sigma \lambda_k (p^{(k)})^T \nabla L_\gamma(\theta^{(k)}) \quad (2.12)$$

holds. We chose  $\sigma = 10^{-2}$ , meaning we accept a decrease in  $L_\gamma$  by 1% of the prediction based on the linear extrapolation.

Now, instead of computing  $H^k$  anew in every iteration, it is merely updated to account for the curvature measured in the most recent step. Given a new iterate  $\theta^{(k+1)}$ , the update for  $H^{(k)}$  in the BFGS approach is (Nocedal and Wright, 2006, Chap. 6.1)

$$H^{(k+1)} = (V^{(k)})^T H^{(k)} V^{(k)} + \rho^{(k)} s^{(k)} (s^{(k)})^T, \quad (2.13)$$

where

$$\rho^{(k)} = \frac{1}{(y^{(k)})^T s^{(k)}}, \quad V^{(k)} = I - \rho^{(k)} y^{(k)} (s^{(k)})^T,$$

and

$$s^{(k)} = \theta^{(k+1)} - \theta^{(k)}, \quad y^{(k)} = \nabla L_\gamma(\theta^{(k+1)}) - \nabla L_\gamma(\theta^{(k)}).$$

This update has the property, that if  $H^{(k)}$  is positive definite and the *curvature condition*

$$(y^{(k)})^T s^{(k)} > 0 \quad (2.14)$$

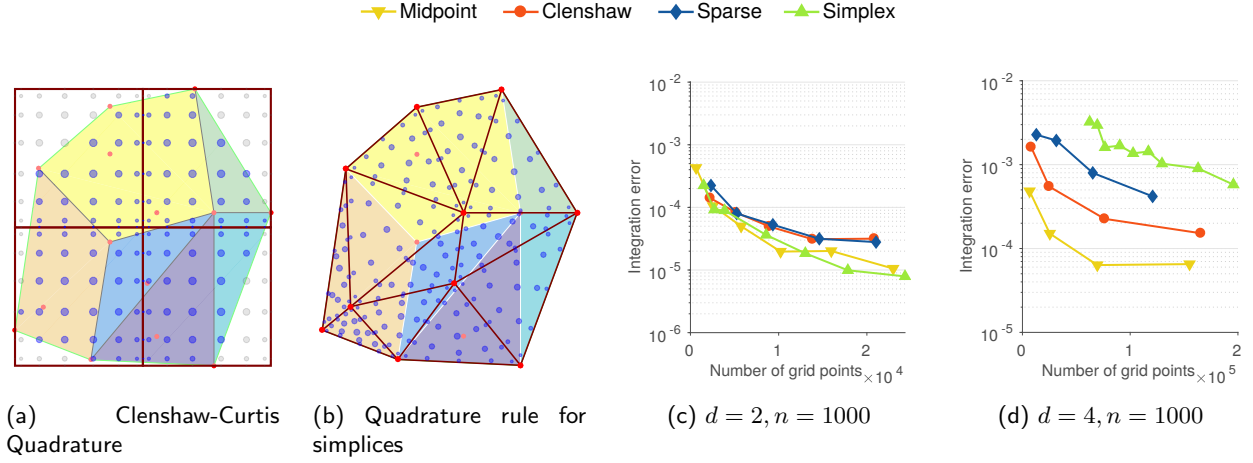
is fulfilled, then  $H^{(k+1)}$  is also positive definite, which in turn guarantees that the step  $p^{(k+1)}$  (2.11) is a descent direction.

While (2.14) automatically holds in the convex case, this property has to be enforced explicitly for non-convex objective functions. Li and Fukushima (2001), therefore, proposed the following modification of  $y^{(k)}$ :

$$\tilde{y}^{(k)} = y^{(k)} + t_k s^{(k)}, \quad t_k = \|\nabla L_\gamma(\theta^{(k)})\| + \max \left\{ -\frac{(y^{(k)})^T s^{(k)}}{\|s^{(k)}\|^2}, 0 \right\}, \quad (2.15)$$

which fulfills (2.14) since  $(\tilde{y}^{(k)})^T s^{(k)} \geq \|\nabla L_\gamma(\theta^{(k)})\| \|s^{(k)}\|^2 > 0$ . Thus using  $\tilde{y}^{(k)}$  in (2.13) guarantees the positive definiteness of  $H^{(k+1)}$ . See Li and Fukushima (2001, Thm. 5.1) for a proof of convergence.

Storing  $H^{(k)}$  in memory quickly becomes prohibitive with growing  $\dim(\theta)$ . This is addressed by *limited-memory* BFGS (L-BFGS) by only storing the  $m$  most recent vectors  $(y^{(k)}, s^{(k)})$ , representing  $H^{(k)}$  implicitly. At every iteration  $p^{(k)}$  (2.11) is directly calculated by recursively applying formula (2.13), see (Nocedal and Wright, 2006, Ch. 7.2). As a result, this approximation of  $H^{(k)}$  only requires the curvature information of the last  $m$  steps. We set  $m = 40$  to obtain a reasonably accurate approximation.



**Figure 2.1:** (a-b) Two integration schemes for the dataset from Figure 1.1 illustrate difficulties that arise for convex integration areas: Integration schemes (a) designed for cubic integration areas lose accuracy when truncated outside the convex integration domain (gray dots). Schemes (b) suited for simplicial integration domains degrade when the simplices are not well aligned to the polyhedral subdivision of  $C_n$  induced by  $\varphi_n(x)$  (1.5). Higher dimensions  $d$  aggravate these effects. (c-d) Simple Riemann sums using the midpoint rule and uniform weights performed best in our experiments, as they do not assume any specific integration area. Simplex based schemes worked only well for small  $n$  and  $d$ .

### 2.3. Numerical Integration

The numerical optimization steps of Section 2.2 require the accurate integration of smooth functions over  $C_n$ , due to (2.7).

We examined various numerical integration schemes: Sparse grid approaches (Bungartz and Griebel, 2004) utilize truncated tensor products of one-dimensional quadrature rules and scale well with the dimension  $d$ . But they are not practical for integrating over non-quadrilateral domains (Bungartz and Griebel, 2004, Sec. 5.3), an observation confirmed by our experiments. Another family of approaches are Monte-Carlo methods based on random-walks (Lovász and Vempala, 2006; Rudolf, 2013), that specifically address the problem of integrating a log-concave density  $f$  over a convex body. Nevertheless, experimental results (Lovász and Deák, 2012) raised doubts about their efficiency, and we did not further pursue this type of approach.

In addition, we examined various dense integrations schemes for the hypercube (simple Newton-Cotes schemes and Clenshaw-Curtis quadrature) as well as schemes tailored to simplicial integration domains, e.g. Grundmann and Möller (1978). Again, regarding the quadrature rules designed for the hypercube, the need to truncate them outside convex subsets (illustrated in Figure 2.1 (a)) had a negative impact on integration accuracy. On the other hand, the simplex-based schemes only worked well if the randomly chosen simplicial decomposition of  $C_n$  for the integration resembled the decomposition (1.4) of  $\hat{\varphi}_n$ . This was only the case for small  $d$  and  $n$ , however. Figure 2.1 (b) illustrates a typical case of misalignment.

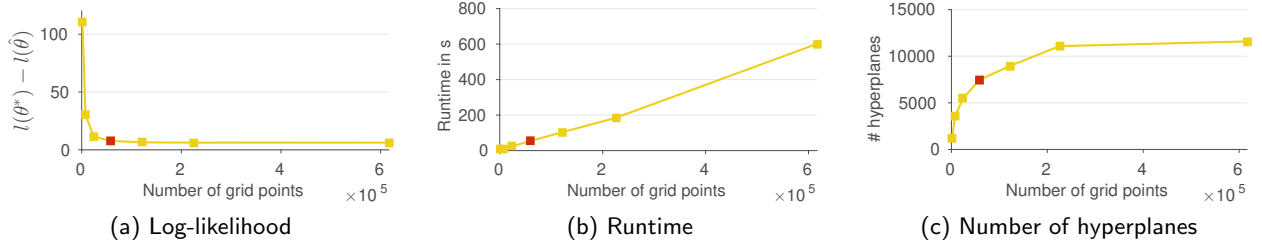
Overall, simple Riemann sums with uniform integration weights performed best in our experiments, because the influence of the shape of the integration domain is minor (Figure 2.1 (c-d)). For future reference, we denote the integration grid by

$$\mathcal{Z}_m = \{z_1, \dots, z_m\} \subset \mathbb{R}^d, \quad (2.16)$$

and the uniform integration weights by  $\Delta$ . Accordingly, the numerical approximation of the integral of the objective (2.7) reads

$$\int_{C_n} \exp(-\varphi_{n,\gamma}(x)) dx \approx \Delta \sum_{i=1}^m \exp(-\varphi_{n,\gamma}(z_i)). \quad (2.17)$$

While naively evaluating (2.17) for the combination of all hyperplanes and grid points would quickly become intractable, we point out that the impact of each hyperplane  $(a_i, b_i)$  is close to zero for most grid



**Figure 2.2:** Effect of the grid density (number of grid points of  $\mathcal{Z}_m$ ) versus (a) quality, (b) runtime and (c) complexity of the solution for a sample of  $n = 5000$  points in  $\mathbb{R}^4$ . As the log-likelihood converges rapidly along with the number of hyperplanes, the runtime increases linearly with the number of grid points. The red marked square defines the density used in our implementation, a trade-off between accuracy and runtime.

points. This fact combined with further plausible measures renders the integration task tractable even for larger dimensions. See [Appendix A](#) for details and discussion.

Regarding the *density* of the integration grid, we traded off accuracy against computational complexity. Figure 2.2 (a) illustrates, for a sample of 5000 points in  $\mathbb{R}^4$ , how the accuracy improves with increasing grid density and finally converges to a solution close to the optimum. As expected, the runtime grows linearly with the number of grid points (Figure 2.2 (b)). In general we found the ratio of grid points to number of hyperplanes (Figure 2.2 (c)) to be a good performance indicator. Keeping this ratio above 3 yielded good results, and we set a minimal number of grid points based on the expected number of hyperplanes for each dimension  $d$  (except for 6-D, where we chose a lower ratio for performance reasons). The red square indicates our choice for 4-D.

#### 2.4. Initialization

Initialization of  $\theta$  plays a crucial role due to the non-convexity of  $L_\gamma(\theta)$ . We examined two different approaches: The *first approach* is based on a kernel density estimate  $f_{\text{kernel}}(x)$  as in [Cule et al. \(2010\)](#), using a multivariate normal kernel with a diagonal bandwidth matrix  $M$  with entries

$$M_{jj} = \sigma_j n^{-1/(d+4)}, \quad j = 1, \dots, d,$$

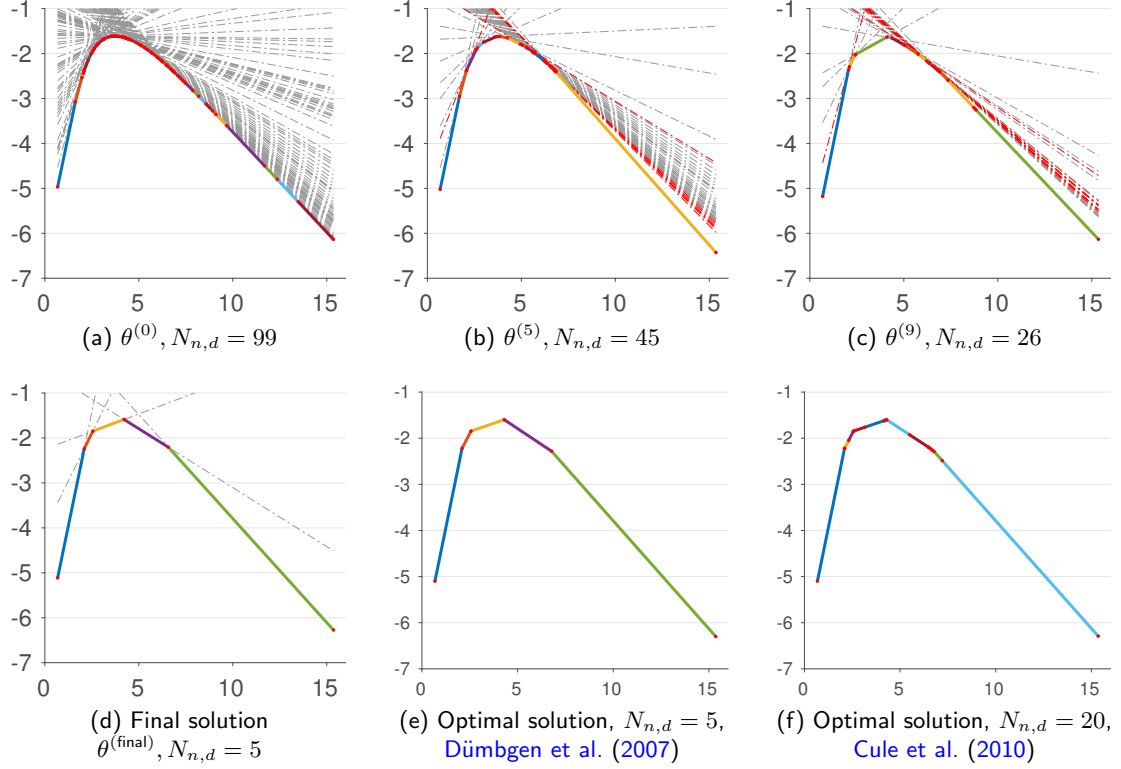
where  $\sigma_j$  is the standard deviation of  $\mathcal{X}_n$  in dimension  $j$ . Setting  $y_i = \log f_{\text{kernel}}(x_i)$  for  $i = 1, \dots, n$ , we compute a simplicial decomposition of  $C_n$  induced by the upper convex hull of  $(X, y)$ , using the popular quickhull algorithm ([Barber et al., 1996](#)). The simplicial decomposition combined with  $y$  then yields an initial set of hyperplane parameters  $\theta^{(0)}$ , one for each simplex  $C_{n,i}$ .

As for the *second approach*, we randomly initialize a small number of hyperplanes and optimize  $L_\gamma(\theta)$  with  $\gamma = 1$ . The rationale behind this is that since  $\gamma$  governs the degree of non-convexity and smoothness of  $L_\gamma(\theta)$ , its optimization is less involved than for smaller  $\gamma$ . Having found the optimal log-concave density for  $\gamma = 1$ , we evaluate  $y_i$  for all  $x_i$  and proceed as described above (first approach) to obtain  $\theta^{(0)}$ . Regarding the specific choice for  $\gamma$ , experiments showed that initializations with  $\gamma = 1$  yielded superior results compared to other initial values of  $\gamma$ , thus offering the “best” trade-off between smoothness of the objective and initial accuracy of the max approximation.

Except for small datasets, in general the second initialization performs better. In practice, we calculate both and select the one with smaller  $L_\gamma(\theta^{(0)})$ .

#### 2.5. Pruning Inactive Hyperplanes

Both initializations produce a very large set of hyperplanes based on a simplicial decomposition of  $C_n$ , with one hyperplane per simplex. During the optimization, hyperplanes may become inactive. Inactivity of some hyperplane  $(a_j, b_j)$  in the light of (1.5) means that there exists no  $x \in C_n$  for which  $\hat{\varphi}_n(x) = \langle a_j, x \rangle + b_j$ . In terms of our smooth approximation  $\varphi_{n,\gamma}(x)$  (2.6), every hyperplane contributes due to  $\exp(x) > 0, \forall x \in \mathbb{R}$ ,



**Figure 2.3:** (a)-(d) Several steps of the optimization process for a sample of 100 points in  $\mathbb{R}$  drawn from a gamma distribution  $\text{Gamma}(\alpha = 5, \beta = 1)$ . Hyperplanes to be dropped in the next step are drawn in red. (a) The initial density is represented by 99 hyperplanes, one for each simplex of the simplicial decomposition of  $C_n$  and based on the smooth max-approximation  $\varphi_{n,\gamma=1}(x)$  (see Section 2.6). Plots (b) and (c) show the transition towards the optimal final shape composed of  $N_{n,d} = 5$  hyperplanes (d). For comparison the non-sparse solution of Cule et al. (2010) (f) and in (e) the optimal solution (only available in 1-D) with a *minimal* representation by Dümbgen et al. (2007) which is identical to the solution returned by our approach.

albeit the contribution may be very close to 0. We therefore resort to the following definition of inactivity using our integration grid:

$$\sum_{i=1}^m \frac{\exp(\gamma^{-1}(\langle a_j, z_i \rangle + b_j))}{\sum_{k=1}^{N_{n,d}} \exp(\gamma^{-1}(\langle a_k, z_i \rangle + b_k))} \leq \vartheta. \quad (2.18)$$

After each update of  $\theta^{(k)}$  we remove all hyperplanes that satisfy (2.18) with  $\vartheta = 10^{-3}$ , which corresponds to a total contribution of less than  $10^{-3}$  grid points. We chose this criterion after observing that hyperplanes usually remained inactive once they lost support on the integration grid, due to their very small contribution to the objective function gradient.

Figure 2.3 visualizes several intermediate steps during an optimization process for  $d = 1$ , together with the shrinking set of hyperplanes. Plots (d)-(f) show the effectiveness of this scheme, since we arrive at the same parametrization as the approach of Dümbgen et al. (2007), which – provided  $d = 1$  – finds the *minimal* representation for  $\hat{\varphi}_n(x)$  in  $\mathbb{R}$ .

### 2.6. Termination

To determine convergence at each iteration, we check if the following inequalities are satisfied:

$$|1 - \Delta \sum_{i=1}^m \exp(-\hat{\varphi}_{n,\gamma}(z_i))| \leq \epsilon, \quad (2.19a)$$

$$|L_\gamma(\theta^{(k+1)}) - L_\gamma(\theta^{(k)})| \leq \delta, \quad (2.19b)$$

where we use  $\epsilon = 10^{-3}$  and  $\delta = 10^{-7}$ . The first criterion asserts that the current density estimate  $\hat{f}_n = \exp(-\hat{\varphi}_{n,\gamma})$  satisfies  $\int \hat{f}_n dx \geq 1 - \epsilon$ . Then second condition detects when the decrease of the objective function becomes negligible. We denote the final parameter vector by  $\theta^{(\text{final})}$ .

### 2.7. Exact Normalization

As a final step, after convergence of the optimization algorithm, we normalize the estimated density using *exact* integration and the non-smoothed representation (1.7) of  $\hat{\varphi}_n$ , which may be seen as setting  $\gamma = 0$  in (2.6). Setting  $y_i = \hat{\varphi}_{n,\gamma}(x_i)$  for all  $x_i \in \mathcal{X}_n$ , we again use `qhull` (Barber et al., 1996) to obtain a triangulation of  $C_n$  and calculate a hyperplane  $\hat{\varphi}_{i,n}$  for every simplex  $C_{n,i}$ . We then split the integral over  $C_n$  into separate integrals over simplices  $C_{n,i}$  and denote the result by  $\lambda$ :

$$\int_{C_n} \exp(-\hat{\varphi}_n(x)) dx = \sum_{i=1}^{N_{n,d}} \int_{C_{n,i}} \exp(-\hat{\varphi}_{i,n}(x)) dx := \lambda \quad (2.20)$$

We make use of Lemma Appendix B.1 to evaluate (2.20) exactly.

The value of  $\lambda$  is close to 1 but *not equal* to 1. We therefore add the same offset parameter  $\delta$  to every hyperplane  $\hat{\varphi}_{i,n}$ , to obtain

$$\tilde{\varphi}_{i,n}(x) := \hat{\varphi}_{i,n}(x) + \delta = \langle x, a_i \rangle + b_i + \delta, \quad i = 1, \dots, N_{n,d}. \quad (2.21)$$

Inserting  $\tilde{\varphi}_{i,n}$  into (B.1) shows that the integral for the modified hyperplanes (2.21) changes to  $\exp(-\delta)\lambda$ . Therefore, after setting  $\delta = \log(\lambda)$ , the final density estimate is  $\hat{f}_n = \exp(-\tilde{\varphi}_n)$  with  $\tilde{\varphi}_n|_{C_{n,i}} = \tilde{\varphi}_{i,n}$  given by (2.21). We denote the corresponding *dense* parameter vector by

$$\hat{\theta}. \quad (2.22)$$

The normalization process eliminates the sparsity of the parametrization used for optimization, since it relies on a simplicial decomposition of  $C_n$ , as does the approach of Cule et al. (2010). Nevertheless, the results of our approach are shown using the sparse parameterization  $\theta^{(\text{final})}$ , which is the essential characteristic of our approach causing the significant speed ups reported in Section 3, whereas the reported log-likelihood values are for  $\hat{\theta}$ .

## 3. Experiments

This section provides empirical evidence that our approach can find sparse solutions that are very close to the optimum and determined computationally using substantially less CPU runtime. Section 3.1 contrasts qualitative properties of our approach with the state of the art, using experiments in dimensions  $d \in \{1, 2\}$  along with illustrations. Quantitative results up to dimension  $d = 6$  are reported in Section 3.2. Finally, we extend our approach to the estimation of mixtures of log-concave densities in Section 3.3.

For all our experiments we used  $\gamma = 10^{-3}$  for determining the accuracy of the smooth approximation  $\varphi_{n,\gamma}$  (2.6) to the max function. This choice is a compromise between less accurate approximations (that is larger gammas) and more accurate but numerically unstable ones. As noted in the previous section, qualitative results show  $\theta^{(\text{final})}$ , i.e. the sparse parametrization found during the optimization (Section 2.2 - 2.6), whereas quantitative results are reported in terms of  $\hat{\theta}$ , the *dense* parametrization obtained by

---

**Algorithm 1:** Fast Log-Concave Density Estimation

---

**Input:**  $X$ , parameters:  $\gamma = 10^{-3}, \vartheta = 10^{-3}, \epsilon = 10^{-3}, \delta = 10^{-7}$   
**Output:** Log-concave density estimate  $\hat{f}_n$  parametrized by  $\theta$  (2.1).  
Find initial  $\theta^{(0)}$  (Section 2.4);  
**for**  $k = 1, 2, \dots$  **do**  
    Delete inactive hyperplanes from  $\theta^{(k)}$  based on criterion (2.18);  
    Compute the gradient  $\nabla L_\gamma(\theta^{(k)})$  of the objective (2.7) using numerical integration;  
    Find descent direction  $p^{(k)}$  from the previous  $m$  gradients vectors and step size  $\lambda_k$  (2.11) and update  $\theta^{(k+1)}$ ;  
    **if** the termination criterion (2.19) holds, **then**  
        Denote final parameter vector by  $\theta^{(\text{final})}$ ;  
        Quit for-loop;  
    **end**  
**end**  
Switch from  $\hat{\varphi}_{n,\gamma}$  to  $\hat{\varphi}_n$  and perform exact normalization:  $\theta^{(\text{final})} \rightarrow \hat{\theta}$  (Section 2.7);  
**return**  $\hat{\theta}$  (2.22)

---

performing the final analytical normalization (Section 2.7), and the non-smoothed representation (1.7) of  $\varphi_n$ . The quality of our solutions is measured in terms of

$$l(\hat{\theta}) = \sum_{i=1}^n -\varphi_n(x_i), \quad (3.1)$$

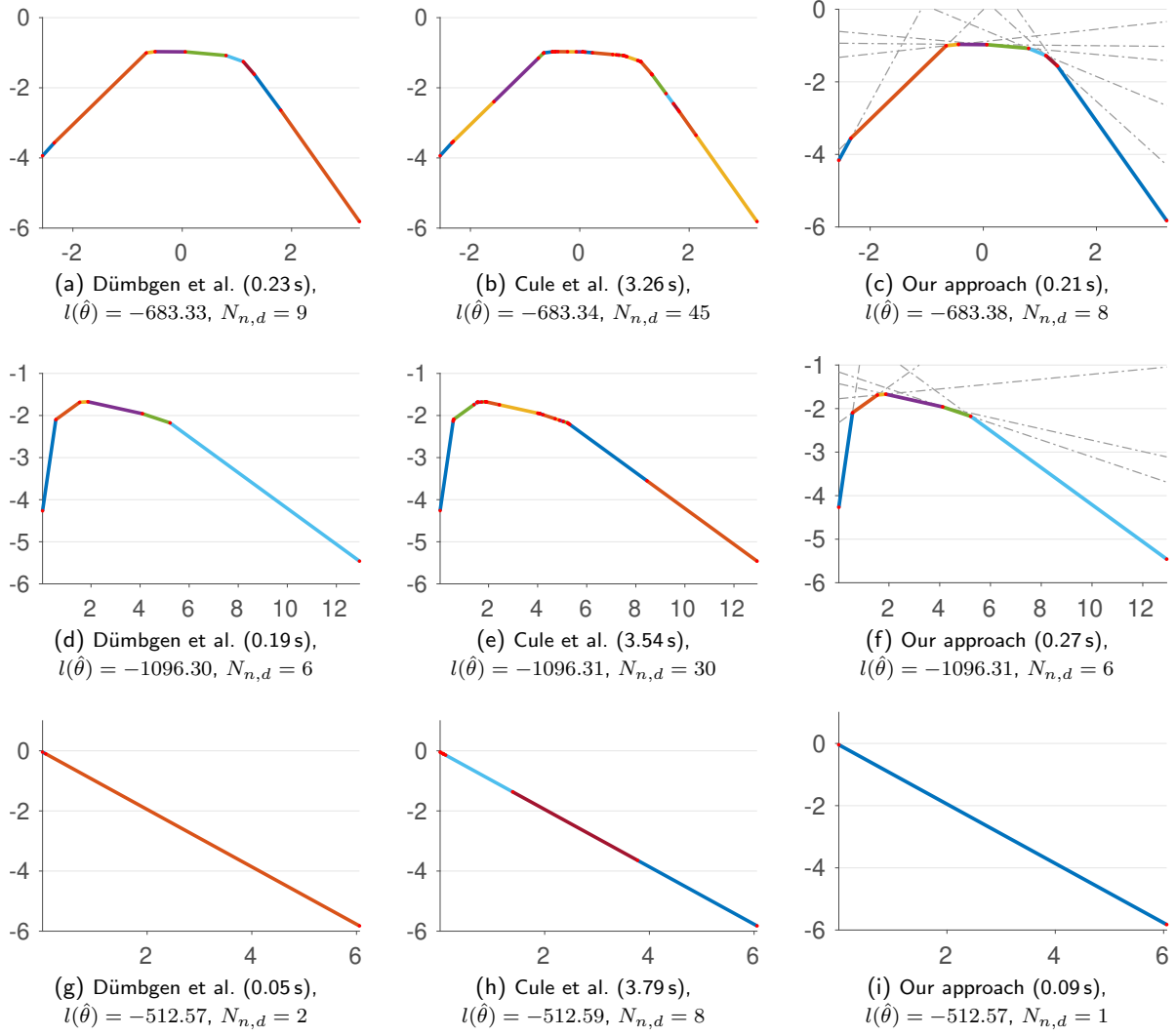
the total log-likelihood.

### 3.1. Qualitative Evaluation

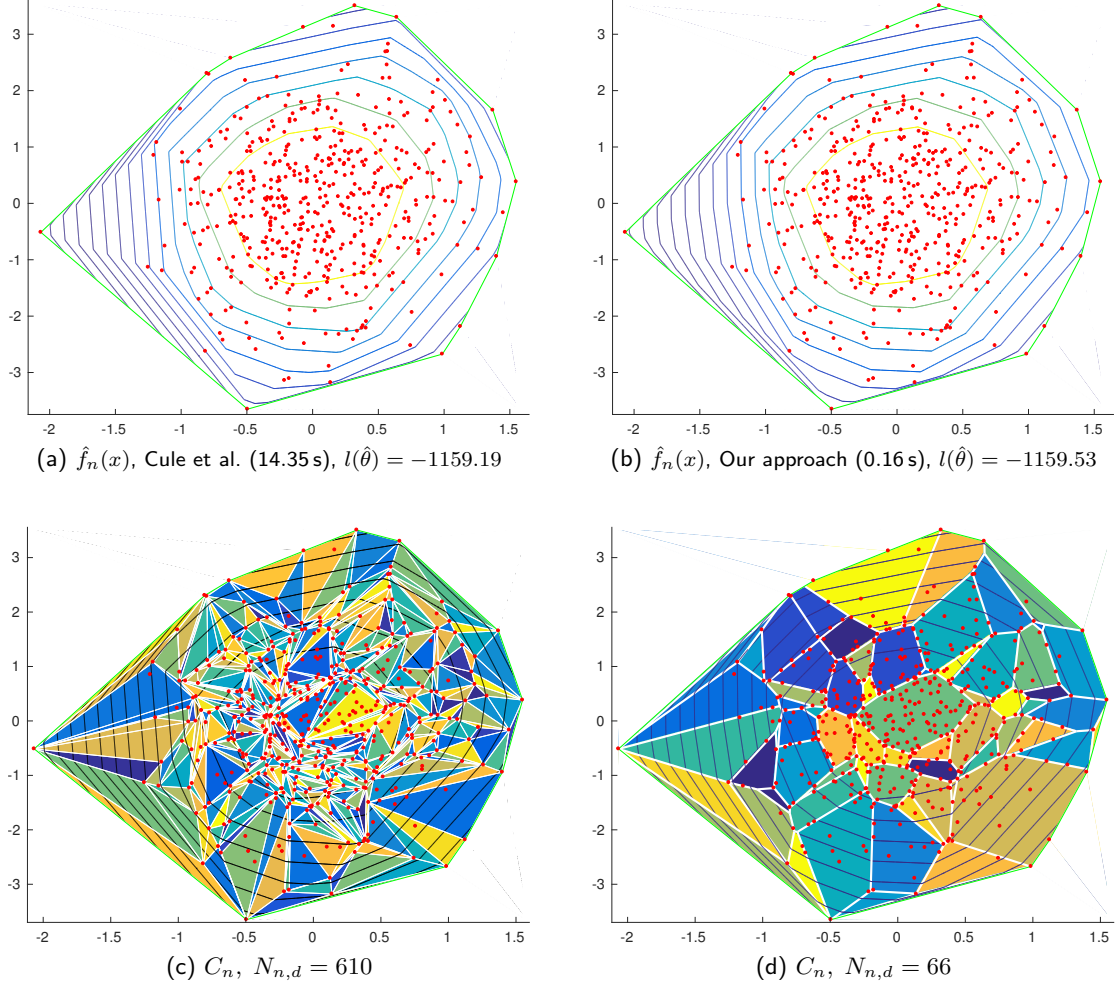
In order to illustrate the sparse structure of solutions determined by our approach, we investigated examples in one and two dimensions. We compared the results with Cule et al. (2010) whose approach (implemented in the R package `LogConcDEAD`) finds optimal solutions in terms of  $L(\theta)$ , but cannot take advantage of any sparsity of the solution, since its representation is based on the fixed triangulation induced by  $\mathcal{X}_n$  (recall Figure 1.2). For  $d = 1$ , we additionally compared to the approach of Dümbgen et al. (2007) using their R package `logcondens`, which can be only applied to univariate data, but finds optimal solutions in terms of  $L(\theta)$  and utilizes the minimal representation of the solution in terms of the number  $N_{n,d}$  of hyperplanes. We sampled 500 data points from three different distributions in 1-D (normal, gamma and exponential) and 500 samples from a normal distribution  $\mathcal{N}(0, I_2)$  in 2-D.

Figure 3.1 depicts the results of all three approaches for univariate data in terms of  $-\hat{\varphi}_n(x) = \log \hat{f}_n(x)$ . While all solutions are almost identical in terms of the estimated density, their parametrizations differ. The solution of Dümbgen et al. (2007) is guaranteed both to have the optimal sparse structure in terms of the number of hyperplanes and to yield the optimal value  $L(\hat{\theta})$ . Comparing their solution to ours, we see that they are almost identical, with only hyperplanes missing that have very small support and negligible impact on  $L(\hat{\theta})$  and  $l(\hat{\theta})$  respectively. The solution of Cule et al. (2010), on the other hand, is densely parametrized. The runtimes reflect these different parametrizations.

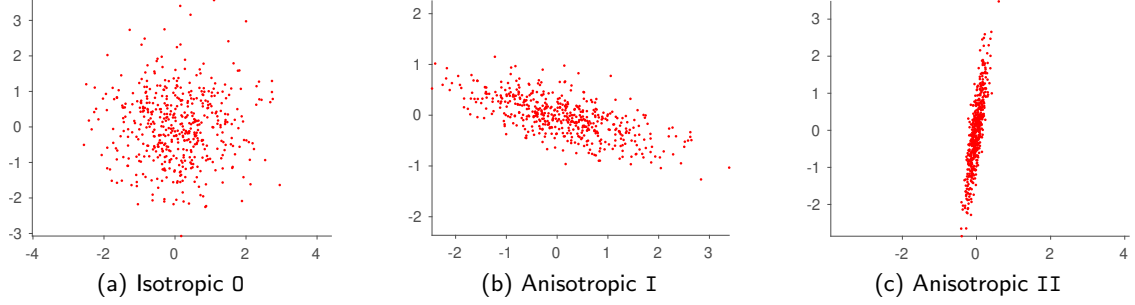
We made similar observations for two-dimensional data  $d = 2$ . Our approach found an density estimate  $\hat{f}_n$  that is almost identical to the optimal solution but required only about 10% of the parameters. Panels (a) and (b) of Figure 3.2 depict the density  $\hat{f}_n(x)$  estimated by the approach of Cule et al. (2010) and our approach, respectively, whereas panels (c) and (d) show the respective decompositions of  $\hat{\varphi}_n(x)$  into its affine representation  $\hat{\varphi}_{i,n}$  (1.5). While the solution of Cule et al. (2010) is based on a fixed hyperplane arrangement and simplicial supports  $C_{n,i}$  induced by the given data, our decomposition of  $C_n$  (1.4) is adaptive and can utilize more general convex polytopes  $C_{n,i}$ . Comparing panels (c) and (d) of Figure 3.2 clearly shows the



**Figure 3.1:** Estimates  $-\hat{\varphi}_n(x)$  and their piecewise linear representation for 500 samples drawn from a (a-c) normal distribution  $\mathcal{N}(0, 1)$ , (d-f) gamma distribution  $\Gamma(2, 2)$  and (g-i) exponential distribution with  $\lambda = 1$ . (a,d,g) The approach of Dümbgen et al. (2007) returns ground truth for univariate data, that is the maximal log-likelihood  $l(\hat{\theta})$  and the correct piecewise linear representation of  $-\hat{\varphi}_n$ . (b,e,h) The approach of Cule et al. (2010) also returns the optimal value  $l(\hat{\theta})$  but generally works with a redundant piecewise linear representation that significantly increases runtime. (c,f,i) Our approach is as efficient as the former and only misses components of the piecewise linear representation that have very small support and hence a negligible impact on  $l(\hat{\theta})$ .



**Figure 3.2:** *Top row:* Contour lines displaying estimates  $\hat{f}_n(x) = \exp(\hat{\varphi}_n(x))$  for a sample of size  $n = 500$  drawn from  $\mathcal{N}(0, I_2)$ . The density plots (a) and (b) as well as the log-likelihood values  $l(\hat{\theta})$  demonstrate that both solutions are very close to each other. *Bottom row:* Decomposition of  $C_n$  induced by the piecewise-linear concave functions  $-\hat{\varphi}_n(x)$ . While the approach of Cule et al. induces a triangulation of  $C_n$ , our approach works with more general polytopes. Our approach adapts this representation to given data and thus avoids overfragmented representations as depicted by (c) that significantly increase runtime.



**Figure 3.3:** The three levels of anisotropy used in our experiments: None (a), mild (b) and strong (c).

beneficial effect of adaptivity which is an ‘automatic’ byproduct of minimizing  $L(\theta)$  using our approach, together with pruning inactive hyperplanes.

### 3.2. Quantitative Evaluation

Besides the introductory experiments used for illustration, we conducted a comprehensive numerical evaluation using more challenging examples. The authors of [Cule et al. \(2010\)](#) reported the evaluation of up to  $n = 2000$  data points and dimension  $d = 4$ , drawn from  $\mathcal{N}(0, I_d)$ . In order to demonstrate the ability of our approach to process efficiently large data sets, we evaluated samples set of size up to  $n = 10000$  points and dimension  $d = 6$ , drawn from the same distribution.

We extended their benchmarks in terms of sample size, dimension as well as introducing anisotropy to the covariance matrices: Besides the identity matrix (degree 0), we defined two levels of anisotropy (I, II) based on the following metric for symmetric positive definitive matrices  $S_1$  and  $S_2$  ([Bhatia, 2006](#)):

$$d(S_1, S_2) = \sqrt{\sum_{i=1}^d \left( \log \lambda_i(S_1^{-1} S_2) \right)^2}. \quad (3.2)$$

Setting  $S_1 = I_d$ , we can see that the metric reduces to the euclidean norm of the logarithm of eigenvalues of  $S_2$ . We define our two levels of anisotropy as

$$\begin{aligned} \text{I} : d(I_d, S_2) &= 2.5, \\ \text{II} : d(I_d, S_2) &= 5.0. \end{aligned} \quad (3.3)$$

Let  $S_2$  be

$$S_2 = DED^T, \quad E = \text{diag}(e), \quad (3.4)$$

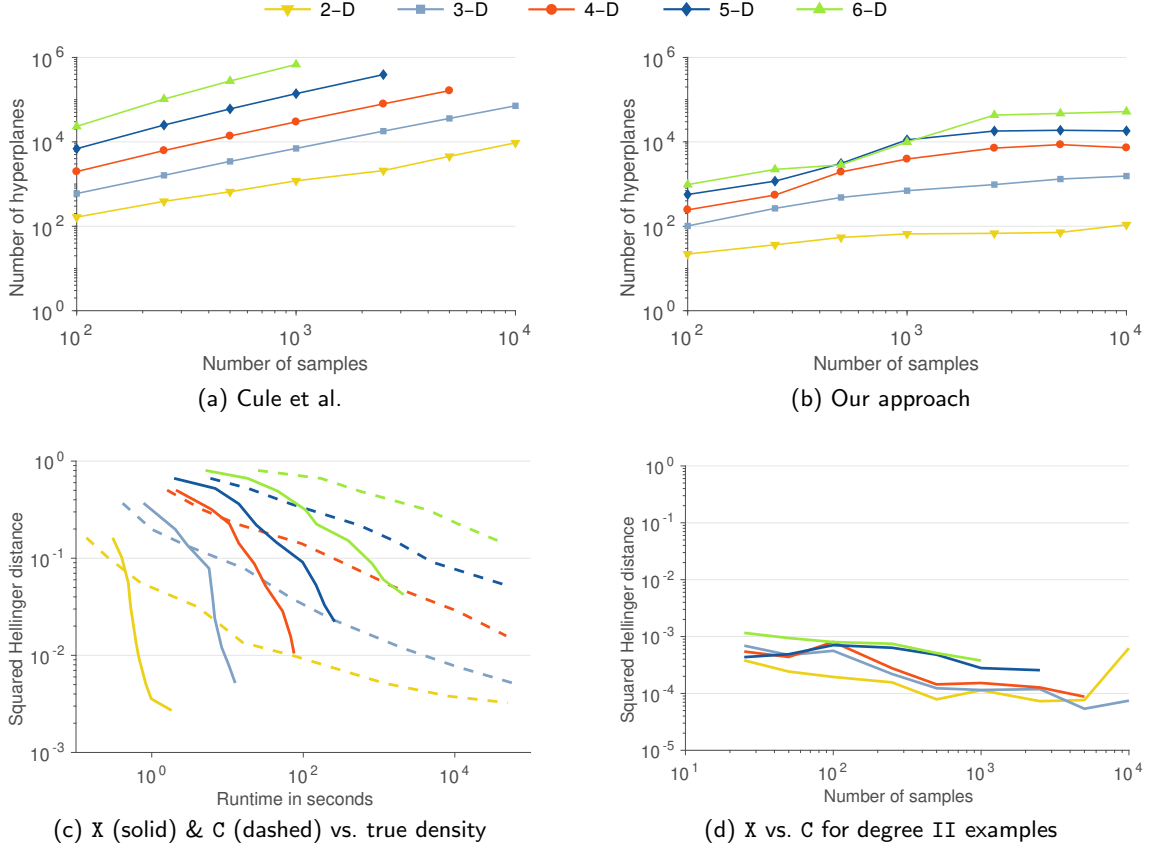
with  $D$  being a *random* orthogonal matrix. Since the eigenvalues of  $S_2$  are solely determined by the vector  $e$ , we fix the last value of  $e$  to 1 and set the remaining values to be equidistant in log-space, such that  $d(I_d, S_2) = \|\log e\|_2 \in \{2.5, 5.0\}$ . For dimension  $d = 2$  and  $n = 500$ , Figure 3.3 depicts examples for each degree of anisotropy.

For each level of anisotropy, we drew samples of sizes  $n \in \{100, 250, 500, 1000, 2500, 5000, 10000\}$  in dimensions  $d \in \{2, \dots, 6\}$  and repeated each experiment five times. We run the R package `LogConcDEAD` (Version 1.6-1) by [Cule et al. \(2010\)](#) in all dimensions but stopped increasing  $n$  when the runtime exceeded 24 hours. Table 3.1 reports the results for all sample sizes, the speed ups as well as the quality of the solution achieved by our approach in comparison to the optimal solution returned by the approach of [Cule et al. \(2010\)](#), measured as the difference of total log-likelihoods  $l(\hat{\theta})$ .

Overall we see that our approach delivers very accurate results, with very small differences given the respective number of samples. Regarding the influence of anisotropy on the accuracy, we notice only a slight increase the difference of  $l(\hat{\theta})$ . Figure 3.4 (d) provides a different perspective on the quality of the estimated

**Table 3.1:** Runtimes for Cule's (mark: C) and our approach (X) and the resulting speedups. For  $d = 1$  we additionally report results for the approach of Dümbgen et al. (D). Quality is the difference in total log-likelihood  $l(\hat{\theta})$  to the solution of C. For the multidimensional datasets, accuracy is reported separately for all three levels of anisotropy (0, I, II). While accuracy slightly decreases for examples with strong anisotropy (degree II), it is still comparatively small given the number of samples. For some samples we achieved a better log-likelihood value, because the implementation of Cule et al. terminates after a hard-coded number of 15000 iterations which is no issue with our approach (for 2-D less than 500 iterations are required).

	$n$	100	250	500	1000	2500	5000	10 000
1-D	Runtime	D	0.08 s	0.2 s	0.3 s	0.4 s	0.7 s	2 s
		X	0.02 s	0.02 s	0.03 s	0.03 s	0.06 s	0.1 s
	Speedup	C	0.2 s	0.7 s	3 s	25 s	16 min	1 h 49 min
			10 x	43 x	121 x	780 x	17 695 x	65 513 x
	Quality	D	0.0	0.0	0.0	-0.1	-0.1	-11.2
		X	0.0	0.0	0.1	0.1	0.2	-10.4
2-D	Runtime	X	0.5 s	0.5 s	0.6 s	0.7 s	0.8 s	1.0 s
		C	0.7 s	4 s	16 s	1 min	19 min	2 h 17 min
	Speedup		1 x	8 x	26 x	116 x	1396 x	8358 x
		0	0.1	0.2	0.3	0.7	1.0	1.3
	Quality	I	0.1	0.1	0.4	0.6	1.0	1.2
		II	0.1	0.3	0.8	1.3	1.4	3.3
3-D	Runtime	X	3 s	6 s	6 s	7 s	8 s	10 s
		C	3 s	17 s	1 min	3 min	34 min	2 h 58 min
	Speedup		1 x	3 x	11 x	34 x	248 x	1028 x
		0	0.1	0.3	0.6	1.2	2.3	5.6
	Quality	I	0.3	0.2	0.5	1.1	3.0	5.5
		II	0.6	0.4	0.7	2.0	3.8	9.5
4-D	Runtime	X	11 s	14 s	23 s	32 s	1 min	1 min
		C	13 s	1 min	5 min	24 min	2 h 57 min	13 h 16 min
	Speedup		1 x	7 x	16 x	46 x	200 x	699 x
		0	0.5	0.7	0.5	1.0	2.2	7.6
	Quality	I	0.2	0.9	0.6	0.8	2.0	6.8
		II	0.5	0.7	0.6	3.1	2.5	7.4
5-D	Runtime	X	14 s	24 s	44 s	1 min	2 min	3 min
		C	1 min	9 min	29 min	1 h 23 min	13 h 4 min	-
	Speedup		5 x	23 x	40 x	51 x	319 x	-
		0	0.3	0.8	0.9	1.1	2.8	-
	Quality	I	0.5	0.9	1.0	1.3	3.4	-
		II	0.4	1.4	2.2	2.2	6.4	-
6-D	Runtime	X	46 s	1 min	2 min	6 min	13 min	19 min
		C	9 min	1 h 14 min	3 h 6 min	10 h 8 min	-	35 min
	Speedup		12 x	41 x	75 x	93 x	-	-
		0	0.4	1.4	2.9	2.5	-	-
	Quality	I	0.3	1.4	2.0	3.1	-	-
		II	0.4	1.1	1.6	3.3	-	-



**Figure 3.4:** *Top row:* Complexity of the density  $\hat{f}_n$  in terms of the number  $N_{n,d}$  of linear functions forming  $-\hat{\varphi}_n(x) = \log(\hat{f}_n(x))$ . While scales linearly with the number of samples  $n$  for the approach of Cule et al.  $N_{n,d}$ , our approach shows only sublinear growth and starts to become flat for larger  $n$ . *Bottom row:* Squared Hellinger distance  $h^2(\hat{f}_n, f)$  between (c) the estimate  $\hat{f}_n$  and the underlying density  $f$  versus runtime, evaluated for our test suite introduced in Section 3.2 with additional results for  $n = 25$  and  $n = 50$ . For a specific runtime, our approach (solid lines) obtains estimates closer to the underlying density  $f$  while being able to process significantly more data points  $n$ . Panel (d) depicts the average squared Hellinger distance between our estimates and those of Cule et al. for degree II samples (cf. Figure 3.3). While constituting the most difficult samples, differences are very small and decrease with sample size.

densities, by showing the very small squared Hellinger distance between our estimates and those of C for degree II samples. While our approach estimated almost optimal densities, it achieved speed ups of up to a factor 30 000 (even more for  $d = 1$ , though Cule et al. (2010) point out that their approach is not designed with the one-dimensional case in mind). These factors increase with dimension and, in particular, with the number of data points.

We empirically observed and estimated how runtime  $t$  depends on the sample size  $n$ . Regarding the approach Cule et al. (2010) we observed linear dependency of the number of iterations on  $n$ , as is also mentioned in Cule et al. (2010). Moreover, we found that the complexity of their density estimate  $\hat{f}_n$ , expressed by the number of hyperplanes  $N_{n,d}$ , also grew linearly with  $n$  (see Figure 3.4 (a)), which is in accordance with our discussion in Section 1. Altogether runtime grew quadratically with the number of samples:  $\mathcal{O}(n^2)$ .

Examining the results for our approach revealed that the number of iterations depends much less on  $n$ . For example, for dimension  $d = 3$ , the number of iterations for  $n \in \{100, 1000, 100000\}$  was about the same, while for  $d = 4$  it doubled from  $n = 100$  to  $n = 1000$  but did not change when increasing  $n$  to 10000. This observation relates to the next paragraph below. Concerning the complexity of  $\varphi_n(x)$  depending on

the sample size  $n$ , we found that the number  $N_{n,d}$  of hyperplanes grew sublinearly and started to become flat when  $n > 1000$ , c.f. Figure 3.4 (b). All in all we found the average dependency of  $t$  on  $n$  to be roughly  $O(\sqrt{n})$  for  $d < 6$  and somewhat larger for  $d = 6$ .

The above observations partially relate to computational constraints for dimensions  $d \geq 5$ , where the grid size for numerical integration may limit the number  $N_{n,d}$  of hyperplanes. For example, increasing the number of grid points by the factor 5 left unchanged the number of hyperplanes for  $n = 5000$  and  $d = 2$ , increased it slightly by 20% for  $d = 4$ , but by about 50% for  $d = 6$ . Apparently, this had no impact on the quality of the corresponding density estimates, but we cannot predict  $N_{n,d}$  for  $d > 4$ .

We conducted further experiments in order to demonstrate that our approach achieves both nearly optimal log-likelihoods and short runtimes: Based on the squared Hellinger distance

$$h^2(\hat{f}_n, f) = 1 - \int \sqrt{\hat{f}_n(x)f(x)}dx \quad (3.5)$$

to measure the similarity between the estimate  $\hat{f}_n$  and the true density  $f$ , Figure 3.4 (c) depicts  $h^2(\hat{f}_n, f)$  for the experiments performed in this section, plotted against the required runtime. Additional results for  $n = 25$  and  $n = 50$  are included. The plot demonstrates that our approach obtains much more accurate density estimates within a specific runtime. This enables to take more data points into account than the approach of Cule et al. (2010), due to the adaptive sparse parametrization.

### 3.3. Mixtures of Log-Concave Densities

We extended our approach to the estimation of mixtures of log-concave densities. Let

$$\Pi = \{\pi_1, \dots, \pi_K\}, \quad \sum_{k=1}^K \pi_k = 1 \quad (3.6)$$

be the mixing coefficients for classes 1 to  $K$  and  $\Theta = \{\theta_1, \dots, \theta_K\}$  be class-specific hyperplane parameters. Then the log-concave mixture distribution is

$$f_n(x|\Theta, \Pi) = \sum_{k=1}^K \pi_k f_n(x|\theta_k), \quad (3.7)$$

where  $f_n(x|\theta_k)$  is a log-concave density as defined in Section 1, parametrized by  $\theta_k$ . Given i.i.d samples  $\mathcal{X}_n = \{x_1, \dots, x_n\}$ , maximizing the log-likelihood function

$$L(\Theta, \Pi) := \sum_{i=1}^n \log \sum_{k=1}^K \pi_k f_n(x_i|\theta_k) \quad (3.8)$$

is challenging due to the summation over  $k$  inside the logarithm. A common technique to maximize locally  $L(\Theta, \Pi)$  is the EM algorithm (Dempster et al., 1977). Here one introduces assignment probabilities

$$\gamma_{i,j} = \frac{\pi_j f_n(x_i|\theta_j)}{\sum_k \pi_k f_n(x_i|\theta_k)}, \quad i = 1, \dots, n, \quad j = 1, \dots, K \quad (3.9)$$

that point  $x_i$  belongs to class  $j$  as latent parameters, that are iteratively estimated along with the mixture coefficients and the parameters of the mixture components. More specifically, the EM algorithm iterates the following two steps until convergence: The *E-Step* computes (3.9). The *M-Step* updates the mixing coefficients

$$\pi_k = \frac{1}{n} \sum_i \gamma_{i,k}, \quad k = 1, \dots, K, \quad (3.10)$$

and refines the parameters  $\theta_k$ ,  $k = 1, \dots, K$  by minimizing the modified negative log-likelihood function

$$L_\gamma(\theta_k) := \frac{1}{n_k} \sum_{i=1}^n \gamma_{i,k} \varphi_{n,\gamma}(x_i) + \int_{C_n} \exp(-\varphi_{n,\gamma}(x)) dx, \quad n_k = \sum_i \gamma_{i,k}, \quad k = 1, \dots, K. \quad (3.11)$$

**Table 3.2:** Performance parameters for the estimation of log-concave mixture densities using the approach of [Cule et al. \(2010\)](#) (mark: C) and our approach (mark: X). Estimates based on our approach are very close to the solutions of C in terms of the log-likelihood (*Quality*) as well as the number of misclassified points, in one case even performing significantly better. Moreover, runtime is significantly reduced. Gaussian mixture models (GMMs) are clearly outperformed by log-concave mixtures for both datasets.

		Wisconsin		USPS	
		2-D	3-D	2-D	3-D
Runtime	X	21 s	3 min	38 s	20 min
	C	19 min	1 h 3 min	2 h 21 min	12 h 50 min
<i>Speedup</i>		54 x	20 x	222 x	38 x
<i>Quality</i>		2.6	4.2	1.5	15.3
Missclassified	X	47	47	<b>200</b>	<b>78</b>
	C	<b>45</b>	<b>45</b>	201	109
<i>Quality</i>	GMM	97.18 %	93.36 %	96.88 %	93.93 %
Missclassified	GMM	58	64	239	215

Since the objective function (3.8) is non-convex, good initialization is essential. To obtain initial values for all  $\gamma_{i,j}$ , we follow [Cule et al. \(2010\)](#) and use hierarchical clustering ([Fraley et al., 2012](#)). We terminated the EM approach if the difference between  $L(\Theta, \Pi)$  values in three subsequent iterations dropped below  $10^{-5}$ .

We tested our approach on two datasets:

- The **Wisconsin breast cancer dataset**, consisting of 569 samples with 30 features each, with 357 benign and 212 malignant instances.
- The well-known **USPS dataset**, containing 11000 images of dimension  $32 \times 32$  (i.e. 256 features) of handwritten digits from zero to nine. We selected all samples for the two classes ‘five’ and ‘six’, 1100 each.

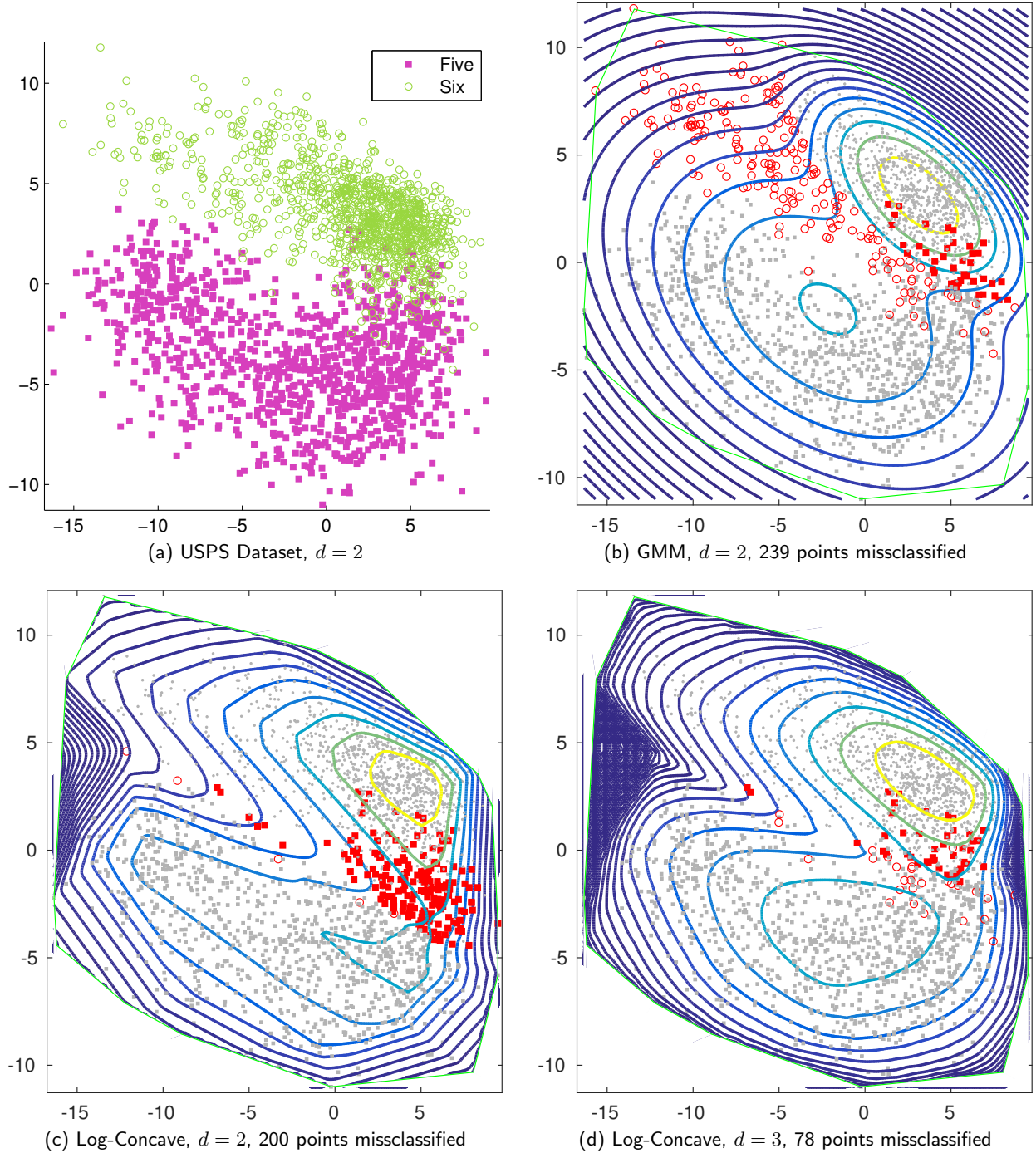
We reduced the dimension for both datasets to  $d \in \{2, 3\}$  using PCA. Figure 3.5 (a) depicts the USPS dataset projected onto the first two PCA eigenmodes. One can see the skewness of class ‘six’, which the Gaussian distribution is not able to capture. We compared our approach to [Cule et al. \(2010\)](#) as well as to the standard Gaussian mixture model (GMM). Performance was measured in terms of the achieved log-likelihood and the number of misclassified samples, where each sample was assigned to the class  $\arg\max_k \pi_k f_n(x|\theta_k)$ .

First, we compare Gaussian mixtures and log-concave mixtures. Table 3.2 demonstrates that the log-concave mixture better reflects the structure of both datasets and clearly outperforms GMMs with respect to both performance measures. Naturally, using more information by increasing the dimension of the PCA subspace may lead to better estimates for the class-wise probabilities, as can be seen for the USPS dataset. Comparing the results of the log-concave mixture density estimates of both approaches, we again see very similar results in terms of the log-likelihood as well as the number of misclassified samples, the only exception being the USPS dataset for  $d = 3$ . Again our approach achieves significant speedups.

Figure 3.5 (b)-(d) visualizes some mixture distributions for the USPS dataset for  $d = 2$  and  $d = 3$ . While the GMM fails to properly model class ‘five’ as expected, log-concave mixtures succeed, especially for  $d = 3$ .

#### 4. Conclusion

This work presented a new computational approach to the estimation of multivariate log-concave densities  $\hat{f}_n$ . Its crucial feature is the iterative search of a sparse representation of  $\hat{f}_n$  in terms of the number of hyperplanes  $N_{n,d}$  comprising the piecewise-linear concave function  $\hat{\varphi}_n(x) = -\log \hat{f}_n(x)$ . In addition, its parametrization involves hyperplanes supported on *general* polytopes  $C_{n,i}$ , whereas the approach of [Cule](#)



**Figure 3.5:** (a) The USPS dataset projected onto the first two eigenmodes. (b,c) GMM and log-concave mixture estimates for this dataset. Misclassified points are drawn red. (d) Log-concave mixture estimate for  $d = 3$  with the third PCA dimension marginalized out by numerical integration for visualization.

et al. (2010) is restricted to simplices. By smoothing the original non-smooth objective, efficient established numerical methods can be applied. Overall, this led to significant speed ups compared to the state-of-the-art approach, in particular for larger data sets and increasing dimension. Although our approach minimizes a non-convex objective, we showed that this does not compromise the quality of the solution. On the contrary, almost optimal results are returned after runtimes that are shorter by factors up to 30 000 x.

Empirical evidence suggests the following dependency on the runtime  $t$  and the sample size  $n$ : We observed that  $t$  grows like  $\mathcal{O}(n^2)$  for the approach by Cule et al. (2010), resulting from a linear growth of both  $N_{n,d}$  and the number of iterations with  $n$ . The observed linear dependency of  $N_{n,d}$  on  $n$  supports our reasoning in Section 1 that the lower bound  $\mathcal{O}(n)$  for  $N_{n,d}$  (cf. Dwyer (1991)) also applies to data sampled from log-concave distributions.

Regarding our approach we observed a  $\mathcal{O}(\sqrt{n})$  dependency, due to a sublinear growth of  $t$  with both  $N_{n,d}$  and the number of iterations. We pointed out that, at least for  $d \leq 4$ , there is strong empirical evidence that the sparse parametrization of  $\hat{\varphi}_n(x)$  reflects structural properties of the maximum-likelihood estimator of log-concave densities, which the approach of Cule et al. (2010) does *not* exploit: Our approach successfully identifies maximal polytopes where  $\log(\hat{f}_n)$  is linear. Since no theoretical results are available, to our knowledge, regarding the sparse parametrization of  $\hat{\varphi}_n(x)$  for the case  $d \geq 2$ , our empirical results may stimulate corresponding research.

We published our code with the R package `fmlogcondens` in the CRAN repository (Rathke and Schnörr, 2018). A Matlab implementation is also available at <https://github.com/FabianRathke/FastLogConvDens>.

#### Acknowledgements

This work has been supported by the German Research Foundation (DFG), grant GRK 1653, as part of the research training group on “Probabilistic Graphical Models and Applications in Image Analysis” <http://graphmod.iwr.uni-heidelberg.de>.

## Appendix A. Calculating $\nabla L_\gamma(\theta^{(k)})$

A large amount of computation time is spent on computing the gradient  $\nabla L_\gamma(\theta^{(k)})$  of the objective (2.7). The gradient component with respect to a hyperplane normal  $a_j$  reads

$$\nabla_{a_j} L_\gamma(\theta) = \frac{1}{n} \sum_{i=1}^n w_{ij} x_i - \Delta \sum_{l=1}^m w_{lj} z_l \exp(-\varphi_{n,\gamma}(z_l)), \quad w_{ij} := \frac{\exp\left(\frac{1}{\gamma}(\langle a_j, x_i \rangle + b_j)\right)}{\sum_{k=1}^{N_{n,d}} \exp\left(\frac{1}{\gamma}(\langle a_k, x_i \rangle + b_k)\right)}. \quad (\text{A.1})$$

Gradient components for the intercepts  $b_j$  are similar. Since  $\frac{1}{\gamma}(\langle a_j, x_i \rangle + b_j) \rightarrow \infty$  for  $\gamma \rightarrow 0$ , a robust evaluation of terms  $w_{ij}$  that prevents numerical overflow of the exp-function is given by

$$w_{ij} = \frac{\exp(\nu_j)}{\sum_{k=1}^{N_{n,d}} \exp(\nu_k)} = \frac{\exp(\nu_j - \max \nu)}{\sum_{k=1}^{N_{n,d}} \exp(\nu_k - \max \nu)}, \quad (\text{A.2})$$

$$\nu_k(x_i) := \gamma^{-1}(\langle a_k, x_i \rangle + b_k), \quad \nu = (\nu_1, \dots, \nu_{N_{n,d}}).$$

Similarly, the smooth max-approximation  $\varphi_{n,\gamma}(x)$  (2.6) is numerically evaluated as

$$\varphi_{n,\gamma}(x) = \gamma \log \sum_{k=1}^{N_{n,d}} \exp(\nu_k) = \gamma \max \nu + \gamma \log \sum_{k=1}^{N_{n,d}} \exp(\nu_k - \max \nu). \quad (\text{A.3})$$

Calculating  $\nabla L_\gamma(\theta^{(k)})$  for all combinations of hyperplanes and grid points is the by far most expensive step in our approach. The problem is inherently sparse, however, since for most grid and data points only a few hyperplanes are relevant with most terms  $w_{ij}$  negligibly small. We exploit this property in several ways.

Computing the exponential function on a CPU is relatively expensive (about 50 times more than addition/multiplication (Ueberhuber, 2012)). We set  $\exp(\nu_j - \max \nu) = 0$ , whenever  $\nu_j - \max \nu \leq -25$ . A second strategy attempts to reduce the number of hyperplane evaluations  $\nu_k$ . It utilizes two integration grids of varying density: A sparse grid to filter inactive hyperplanes and a dense grid to evaluate the integral of  $f(x)$  and its gradient for all active hyperplanes. The sparse grid is divided into boxes  $B_i$  consisting of  $2^d$  adjacent grid points  $\{v_1^i, \dots, v_{2^d}^i\}$ , e.g. 4 boxes in Figure 2.1 (a). For each box  $B_i$  we perform the following steps:

1. Pick the point  $z \in B_i$  that is closest to the mean of  $\mathcal{X}_n$ , evaluate all  $\nu_k(z)$ ,  $k = 1, \dots, N_{n,d}$  and set  $\bar{k} = \arg \max_k \nu_k$ .
2. For each  $k = 1, \dots, N_{n,d}$  find  $\Delta_k$ , the upper bound on the increase of hyperplane  $k$  relative to hyperplane  $\bar{k}$  in  $B_i$ . Let  $w_j$  be the width of box  $B_i$  in dimension  $j$  and

$$\zeta_j = \begin{cases} 1 & z_j = \min_l v_{l,j}^i, \\ -1 & \text{otherwise.} \end{cases} \quad (\text{A.4})$$

Then

$$\Delta_k = \sum_{j=1}^d \max((a_{k,j} - a_{\bar{k},j})w_j \zeta_j, 0). \quad (\text{A.5})$$

3. If  $\nu_k(z) + \Delta_k - \nu_{\bar{k}}(z) \leq -25$ , exclude hyperplane  $k$  from the integration over  $B_i$ .

For medium sized problems, this scheme reduces the number of evaluations of  $w_{ij}$  by about 90%.

Using a numerical integration scheme based on a regular grid facilitates *parallelization*. We automatically distribute the numerical integration (and other expensive for-loops) across all available CPU cores, using the OpenMP API (Dagum and Menon, 1998). In addition, we utilize Advanced Vector Extensions (AVX), a technique that *vectorizes* code by performing certain operations like addition or multiplication simultaneously for 8 floating point or 4 double values on a single CPU core. AVX is supported by all CPUs released since 2010. Both techniques, within-core and across-core parallelization led to speed ups by a factor of more than 10 on a standard four core machine. Due to the local character of most computations, transferring the code to the GPU promises even larger speed-ups.

## Appendix B. Exact integration

Cule et al. (2010) provided an explicit formula in order to evaluate exactly the integral in the objective (1.11) based on a *simplicial decomposition* of  $C_n$ :

**Lemma Appendix B.1.** *Let  $j = (j_0, \dots, j_d)$  be a  $(d+1)$ -tuple of distinct indices in  $\{1, \dots, n\}$ , such that  $C_{n,i} = \text{conv}(x_{j_0}, \dots, x_{j_d})$  is a simplex with associated affine function  $\varphi_{i,n} = \langle a_i, x \rangle + b_i$  and values  $y_{j_l} = \varphi_{i,n}(x_{j_l})$  at its vertices. Then*

$$\int_{C_{n,i}} \exp(-\varphi_{i,n}(x)) dx = \text{vol}(C_{n,i}) \sum_{l=0}^d \exp(-y_{j_l}) \prod_{i \in \{0, \dots, d\} \setminus \{l\}} (y_{j_i} - y_{j_l})^{-1}. \quad (\text{B.1})$$

We apply this Lemma in Section 2.7 in order to normalize exactly the numerically computed density estimate  $\hat{f}_n$  returned by our approach.

## References

- Acharya, J., Diakonikolas, I., Li, J., Schmidt, L., 2017. Sample-optimal density estimation in nearly-linear time. In: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 1278–1289.
- Axelrod, B., Valiant, G., 2018. An efficient algorithm for high-dimensional log-concave maximum likelihood. arXiv preprint arXiv:1811.03204.
- Barber, C. B., Dobkin, D. P., Huhdanpaa, H., 1996. The quickhull algorithm for convex hulls. *ACM T. Math. Software* 22 (4), 469–483.
- Bhatia, R., 2006. Positive Definite Matrices. Princeton Univ. Press.
- Bungartz, H.-J., Griebel, M., 2004. Sparse Grids. *Acta Numer.* 13, 147–269.
- Chen, Y., Samworth, R. J., 2013. Smoothed log-concave maximum likelihood estimation with applications. *Statist. Sinica* 23.
- Cule, M., Samworth, R., Stewart, M., 2010. Maximum likelihood estimation of a multi-dimensional log-concave density. *J. R. Stat. Soc. Series B Stat. Methodol.* 72 (5), 545–607.
- Dagum, L., Menon, R., 1998. Openmp: An industry-standard API for shared-memory programming. *IEEE Comput. Sci. Eng.* 5 (1), 46–55.
- Dempster, A. P., Laird, N. M., Rubin, D. B., 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc. Series B Stat. Methodol.* 39 (1), 1–38.
- Diakonikolas, I., Kane, D. M., Stewart, A., 2017. Learning multivariate log-concave distributions. In: Kale, S., Shamir, O. (Eds.), Proceedings of the 2017 Conference on Learning Theory, PMLR. Vol. 65. pp. 711–727.
- Diakonikolas, I., Sidiropoulos, A., Stewart, A., 2018. A polynomial time algorithm for maximum likelihood estimation of multivariate log-concave densities. arXiv preprint arXiv:1812.05524.
- Dümbgen, L., Hüsler, A., Rüfiba, K., 2007. Active set and EM algorithms for logconcave densities based on complete and censored data. Tech. Rep. 61, University of Bern.
- Dümbgen, L., Rüfiba, K., 2009. Maximum likelihood estimation of a log-concave density and its distribution function: Basic properties and uniform consistency. *Bernoulli* 15 (1), 40–68.
- Dwyer, R. A., 1991. Higher-dimensional voronoi diagrams in linear expected time. *Discrete Comput. Geom.* 6, 343–367.
- Fraley, C., Raftery, A. E., Murphy, T. B., Scrucca, L., 2012. mclust version 4 for R: Normal mixture modeling for model-based clustering, classification, and density estimation.
- Grundmann, A., Möller, H.-M., 1978. Invariant integration formulas for the n-simplex by combinatorial methods. *SIAM J. Numer. Anal.* 15 (2), 282–290.
- Li, D.-H., Fukushima, M., 2001. A modified BFGS method and its global convergence in nonconvex minimization. *J. Comput. Appl. Math* 129 (1), 15–35.
- Lovász, L., Deák, I., 2012. Computational results of an  $O^*(n^4)$  volume algorithm. *Eur. J. Oper. Res.* 216 (1), 152 – 161.
- Lovász, L., Vempala, S., 2006. Fast algorithms for log-concave functions: Sampling, rounding, integration and optimization. In: Foundations of Computer Science (FOCS). IEEE, pp. 57–68.
- Lovász, L., Vempala, S., 2007. The geometry of log-concave functions and sampling algorithms. *Rand. Structures Alg.* 30 (3), 307–358.
- McMullen, P., 1970. The maximum numbers of faces of a convex polytope. *Mathematika* 17 (2), 179–184.
- Nocedal, J., Wright, S., 2006. Numerical Optimization. Springer Science & Business Media.
- Rathke, F., Schnörr, C., 2018. fmlogcondens: Fast multivariate log-concave density estimation. R package version 1.0.2. URL <https://cran.r-project.org/package=fmlogcondens>
- Rockafellar, R. T., Wets, R. J.-B., 2009. Variational Analysis, 3rd Edition. Vol. 317. Springer.
- Rudolf, D., 2013. Hit-and-run for numerical integration. In: Monte Carlo and Quasi-Monte Carlo Methods 2012. Vol. 65. Springer, pp. 597–612.
- Samworth, R. J., 2017. Recent progress in log-concave density estimation. arXiv preprint arXiv:1709.03154.
- Silverman, B., 1982. On the estimation of a probability density function by the maximum penalized likelihood method. *Ann. Stat.* 10 (3), 795–810.
- Ueberhuber, C. W., 2012. Numerical Computation 1: Methods, Software, and Analysis. Springer Science & Business Media.